CSE 331: Introduction to Algorithm Analysis and Design Fall 2012

HOMEWORK 8

Due Friday, November 16, 2012 by 1:15pm in class

Please submit each problem separately, i.e. each problem should begin on a new page and only the pages for one problem should be stapled together. Failure to do so might result in some problem(s) not being graded.

This is probably the toughest HW in the entire course. So please do start EARLY!

Do not turn in Q 0, 4 and 5. They are just for you to get some practice.

---

0. (**DO NOT hand this problem in**) Design an $O(m \log n)$ time algorithm that given an undirected graph $G = (V, E)$ and edge weights $c_e > 0$ for every $e \in E$, outputs the *maximum* spanning tree: i.e. among all the spanning trees for G, the algorithm outputs the one with the maximum total weight of edges (breaking ties arbitrarily).

1. (40 points) In class we have seem algorithms that compute the MST in time $O(m \log n)$. This is under the assumption that the graph is given it its adjacency list representation. This of course is not as best as possible[1].

   In this problem you will explore how to implement Prim's MST algorithm if the graph is given in its adjacency matrix form. (The adjacency matrix for a weighted graph is the natural generalization of the unweighted one we have seen in class earlier. In particular, the entry for (i, j) in the matrix is a ∞ if (i, j) is not an edge, otherwise it is $c_{(i,j)}$.)

   Show how to implement Prim's algorithm in $O(n^2)$ time if the input graph is given as an adjacency matrix.

   Hint: Look at the implementation of Prim's algorithm in the book and try to think how having an adjacency matrix might help you maintain the critical quantity you need to keep track of (for each vertex). You will have to maintain some auxiliary data structure(s) (but simpler one(s) than the one used in the book's implementation).

2. In this problem, we will consider the special case of MST where there are only two possible edge costs. In other words, the input to the problem is an undirected graph $G = (V, E)$ where every edge cost satisfies $c_e \in \{1, 2\}$ (i.e. the edge costs are either 1 or 2 and no other value). Present an $O(m + n)$ time algorithm to compute the MST of such graphs.

   Note: The above run time is better than any known MST algorithm for the general cost case. Thus, like Q2 on the mid-term you will have to do something slightly different.

   Hint: It *might* be useful to start from the $O(m + n)$ time algorithm for the case when $c_e = 1$ for every $e \in E$ and then try to extend the algorithm to the current two cost scenario. In the solution that I have in mind the *proof of correctness* of the algorithm goes via the proof of correctness of Kruskal's algorithm. *As usual, there can be more than one way of solving a problem, so feel free to ignore the hint if you have another way to solve the problem.*

[1] The best known running time for an MST algorithm is $O(m \cdot \alpha(n, m))$, where $\alpha(\cdot, \cdot)$ is the inverse Ackermann function and is a a very slooooooooowly growing function– for all practical input values, the function value is smaller than (say) 5.

3. (15 points) Argue that your algorithm from Q 1 is the best possible: i.e., *no* algorithm that solves the MST problem when the input graph is given in the adjacency matrix format can have a faster asymptotic running time.

   Hint: Recall that the running time of an algorithm has to include the time it takes to write its output and the time to read its input.

4. (**DO NOT hand this problem in**) Exercise 8 in Chapter 4.

5. (**DO NOT hand this problem in**) Exercise 10 in Chapter 4.